TEC | Tecnológico de Costa Rica

Programa del curso IC-5701

Compiladores e Intérpretes

Escuela de Computación

Carrera de Ingeniería de Computación, Plan 411.

I Parte. Aspectos relativos al plan de estudios

1 Datos generales

Nombre del curso:	Compiladores e Intérpretes
Código:	IC-5701
Tipo de curso:	Teórico - Práctico
Nº de créditos:	4
Nº horas de clase por semana:	4
Nº horas extraclase por semana:	8
Ubicación en el plan de estudios:	Curso del V Semestre del Bachillerato de Ingeniería en computación
Requisitos:	IC-4700 Lenguajes de Programación
Correquisitos:	Ninguno
El curso es requisito de:	IC-6600 Principios de Sistemas Operativos y IC-6202 Inteligencia Artificial
Asistencia:	Obligatoria
Suficiencia:	No
Posibilidad de reconocimiento:	Sí
Vigencia del programa:	II semestre 2021.

2 Descripción General

Este curso estudia principios y técnicas necesarios para la construcción de procesadores de lenguajes de programación, con énfasis en compiladores e intérpretes. El aprendizaje permite aplicar modelos abstractos (lenguajes formales y autómatas), enfrentar problemas de manipulación de información simbólica, realizar programación modular avanzada, analizar sistemas complejos de software y hacer modificaciones sistemáticas, y ampliar el repertorio de métodos para resolución de problemas informáticos

3 Objetivos

Objetivo General

Aplicar principios, modelos y técnicas para el diseño y la construcción de procesadores de lenguajes de programación de alto nivel, con énfasis en compiladores e intérpretes.

Objetivos Específicos

- Identificar los principales problemas relacionados con la implementación de lenguajes de programación.
- Comprender principios y métodos para implementar lenguajes de programación.
- Conocer las tareas que deben realizarse para traducir un lenguaje de programación a un lenguaje de máquina.
- Comprender modelos abstractos de lenguajes y máquinas formales (gramáticas, expresiones regulares y autómatas) y su aplicabilidad para la implementación de lenguajes de programación.
- Entender técnicas para construir compiladores e intérpretes que procesen lenguajes procedimentales.
- Diseñar y construir, de manera sistemática, un compilador o/y un interprete para un lenguaje imperativo con estructura de bloques.

4 Contenidos

Introducción a la compilación (1.5 semana)

- Especificación de Lenguajes
- Procesadores de Lenguajes
- Conceptos Generales de la compilación
- Máquinas Reales y Abstractas
- Estructura del Proceso de Compilación
- Bootstrapping

Análisis léxico (2.5 semanas)

- Generalidades del análisis léxico
- Proceso del análisis léxico
- Expresiones regulares
- Automatas (DFA, NFA)
- De Expresiones Regulares a Autómatas y de Autómatas a Expresiones Regulares
- Alternativas de implementación de un analizador léxico
- Tratamiento de errores léxicos
- Construcción automática de analizadores léxicos

Análisis sintáctico (3 semanas)

- Proceso de Análisis Sintáctico
- Gramáticas libres de contexto
- Construcción de árboles sintácticos
- Ambigüedad y recursividad en gramáticas
- Automatas de pila
- Reconocimiento descendente: descenso recursivo, LL(1), First, Follow

- Reconocimiento ascendente:generación de DFA, LR(0), SLR(1)
- Tratamientos de errores sintácticos
- Construccion automática de parsers

Análisis contextual o semántico (1.5 semanas)

- Manejo de alcances de variables y tablas de símbolos
- Validación de tipos.
- Diseño de analizadores de contexto.
- Tratamiento de errores contextuales.

Estructuras en tiempo de ejecución (1.5 semanas)

- Pila Semántica y Registros de activación
- Representacion de datos
- Evaluación de expresiones
- Organización de de memoria durante la ejecución de un programa
- Acceso a objetos no locales
- Rutinas: paso de parámetros, ligas estáticas, argumentos, recusión

Generación de código (3 semanas)

- Estructuras de Datos para generación de código
- Técnicas básicas de generación de código
- Generación de código para Estructuras de Datos
- Generación de código para Expresiones
- Generación de código para Estructuras de control
- Generación de código para Procedimientos y Funciones

Interpretación (1.5 semanas)

- Interpretación recursiva directa.
- Interpretación de un lenguaje imperativo.
- Interpretación de un lenguaje funcional.
- Interpretación iterativa indirecta (de una máquina abstracta).

8 Temas avanzados (1.5 semanas)

- Optimización de código.
- Compilación para arquitecturas paralelas y lenguajes para cómputo de alto rendimiento.
- Compilación para sistemas distribuidos.
- Inferencia de tipos
- Utilización de las técnicas de compilación en otras aplicaciones

5 Aspectos operativos

Metodología de enseñanza y aprendizaje

El profesor desarrollará de manera magistral los aspectos teóricos y prácticos más relevantes de los diferentes temas. Se complementará con otras lecturas, código ejemplo y extractos de libros y manuales relevantes. Se espera una alta participación de los estudiantes durante las lecciones. Se cubrirá en clases mucho del material clásico de compiladores, con lecturas adicionales asignadas a los estudiantes. Se realizarán frecuentes exámenes cortos, que cubren el material teórico y práctico reciente. Se asignarán tareas cortas individuales. La presentación de las tareas debe ser de calidad profesional.

Un componente esencial del aprendizaje y de la evaluación de este curso son los proyectos programados; el profesor indicará en cuáles lenguajes se podrá realizar la programación y si se usarán programas de base o herramientas. Se puede trabajar en grupos de 2 personas. Los grupos de trabajo requieren la aprobación del profesor. Habrá un mínimo de 3 proyectos.

El profesor sugerirá un libro principal de referencia. Se presupone que el alumno profundiza los temas abordados en clase en ese libro y otras lecturas recomendadas por el profesor.

Evaluación

A continuación se detalla la evaluación propuesta del curso:

	%
0. Gramática	15
1. Explorador	15
2. Analizador	15
3. Verificador	15
4. Generador	15
Foros	10
Comprobaciones de vistas	15
	100

El contenido académico de las actividades de mediación será acumulativo, ya que conforme avance el curso se requerirá de los conocimientos previos.

6 Aspectos Administrativos:

- En caso de que se detecte un plagio o intento de fraude en cualquier asignación por parte de un estudiante se procederá según la reglamentación del TEC de acuerdo a lo estipulado en el Reglamento del Régimen de enseñanza aprendizaje Artículo 75.
- No se aceptarán trabajos 10 minutos después de la fecha y hora indicadas. Por lo tanto, trabajos entregados tardíamente tendrán una nota de cero, salvo indicación expresa del votán.
- El curso se aprueba con nota de 70. No hay examen de reposición.
- Todas las asignaciones escritas deben de presentarse en formato **pdf**.
- Todo entregable que contenga más de un archivo deberá ser entregado en una carpeta con el nombre código del grupo de trabajo (para asignaciones grupales) o nombre_apellido (para asignaciones individuales). Esto compreso con el formato **tar.gz**

7 Bibliografía

Los materiales de estudio se proveerán en clase. Sin embargo si desea ampliar por su cuenta la información vista en clase puede recurrir a los siguientes textos.

Referencias principales:

Watt, D.; Brown, D. "Programming language processors in Java". Prentice-Hall, 2000.

Watt, D. "Programming language processors". Prentice-Hall, 1993.

Aho, A.; Sethi, R.; Ullman, J. "Compilers: Principles, techniques and tools". Addison-Wesley, 1986.

Aho, A.; Sethi, R.; Ullman, J. "Compiladores: Principios, técnicas y herramientas". Addison-Wesley Iberoamericana, 1998.

Aho, A.; Lam, M; Sethi, R.; Ullman, J. "Compilers: Principles, techniques and tools", 2nd. Ed . Addison-Wesley, 2006.

Fischer; Le Blanc. "Crafting a Compiler with C". Benjamin Cummings, 1991.

Hopcroft, J.; Motwani,R.; Ullman, J. "Introduction to Automata Theory, Languages and Computation", 3rd. Ed. Addison-Wesley, 2006.

Louden, K. "Compiler construction: Principles and practice". PWS Publishing, 1997.

Referencias complementarias:

Appel, A.; Ginsburg, M. "Modern Compiler implementation in C". Cambridge University Press, 1998.

Backhouse, R. "Syntax of programming languages". Prentice-Hall, 1979.

Barrett et al. "Compiler construction". 2nd. ed. Science Research Associates, 1986.

Calingaert, P. "Assemblers, compilers and program translation". Computer Science Press, 1979.

8 Votán

Ing. Aurelio Sanabria

Aurelio es votán, investigador y activista. Labora en el TEC desde 2014, cuenta chistes de elefantes y tiene un personaje en LOL: Aurelion sol, el forjador de estrellas

Correo: aurelio.itcr@gmail.com
Página web: https://twitter.com/sufrostico

Horario de consulta: De Lunes a Viernes de 8:00 a.m. a 5:00 p.m. de forma asincrónica
Lugar de consulta: Virtual vía Signal a @AnarcoPizote.01